IN THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended): Method for scheduling the service of a thread, said method comprising the steps of:

creating the thread when a first hardware device of hardware devices is initialized, wherein the thread is created for use during processing of a first interrupt that the first hardware device is configured to generate;

freeing the thread when the first hardware device is shut down;

masking interrupts from the hardware devices in order to ignore interrupts for other threads;

acquiring a latency information associated with the thread, wherein the latency information indicates a time at which the thread needs to be processed;

unmasking interrupts from the hardware devices in order to detect interrupts for the other threads; [[and]]

rearranging an order in which the thread and the other threads will be serviced in a single queue to schedule the thread for processing in accordance with said latency information, wherein the rearranging is performed simultaneously for the thread and the other threads, and the thread and the other threads that are ordered in the single queue correspond to all requests received from the hardware devices; and

creating an additional thread, wherein a first interrupt identification number is associated with the thread and a second interrupt identification number that is different than the first interrupt identification number is associated with the additional thread and the additional thread is created for use during processing of a second interrupt that the first hardware device is configured to generate; and

freeing the additional thread when the first hardware device is shut down.

2. (Previously Presented): The method of claim 1, wherein said latency information is computed based on a buffer size or display rate.

3. (Previously Presented): The method of claim 1, further comprising computing the time at which the thread needs to be processed by summing the latency information with a current time.

4. (Previously Presented): The method of claim 1, wherein said latency information represents a time duration that is necessary to service the thread.

5. (Previously Presented): The method of claim 1, wherein said latency information represents a maximum time allowed before a first buffer will be emptied and a read operation will switch to process a second buffer.

6. (Previously Presented): The method of claim 1, wherein said latency information represents a time duration that is necessary to setup the thread to perform interrupt processing for the thread.

7. (Previously Presented): The method of claim 1, wherein said latency information is dependent on a hardware constraint for one of the hardware devices.

8. (Previously Presented): The method of claim 1, wherein said latency information is provided by a device driver.

9. (Currently Amended): Apparatus for scheduling the service of a thread, said apparatus comprising:

means for creating the thread when a first hardware device of hardware devices is initialized, wherein the thread is created for use during processing of a first interrupt that the first hardware device is configured to generate;

means for freeing the thread when the first hardware device is shut down;

means for receiving an interrupt from [[a]]the first hardware device;

means for masking interrupts from the hardware devices in order to ignore interrupts for other threads;

means for acquiring latency information associated with the interrupt, wherein the latency information indicates a time at which the thread needs to be processed;

means for unmasking interrupts from the hardware devices in order to detect interrupts for the other threads; [[and]]

means for rearranging an order in which the thread and the other threads will be serviced in a single queue to schedule the thread to process the interrupt in accordance with said latency information, wherein the rearranging is performed simultaneously for the thread and the other threads, and the thread and the other threads that are ordered in the single queue correspond to all requests received from the hardware devices; and

means for creating an additional thread, wherein a first interrupt identification number is associated with the thread and a second interrupt identification number that is different than the first interrupt identification number is associated with the additional thread and the additional thread is created for use during processing of a second interrupt that the first hardware device is configured to generate; and

means for freeing the additional thread when the first hardware device is shut down.

10. (Previously Presented): The apparatus of claim 9, further comprising computing the time at which the thread needs to be processed by summing the latency information with a current time.

11. (Previously Presented): The apparatus of claim 9, wherein said latency information is dependent on a hardware constraint for one of the hardware devices.

12. (Original): The apparatus of claim 11, wherein said hardware constraint is a size of a buffer.

13. (Original): The apparatus of claim 11, wherein said hardware constraint is a fullness of a buffer.

14. (Previously Presented): The apparatus of claim 11, wherein said hardware constraint is dynamically computed based on a buffer size or display rate.

15. (Original): The apparatus of claim 9, wherein said latency information is generated by a device driver associated with the hardware device.

16. (Canceled)

17. (Previously Presented): The method of claim 1, further comprising toggling an interrupt line.

18. (Previously Presented): The method of claim 1, further comprising:
determining the thread should be activated; and
activating the thread for processing.

19. – 22. (Canceled)

23. (Previously Presented): The method of claim 1, wherein the thread and at least one of the other threads correspond to interrupt requests from a single one of the hardware devices.